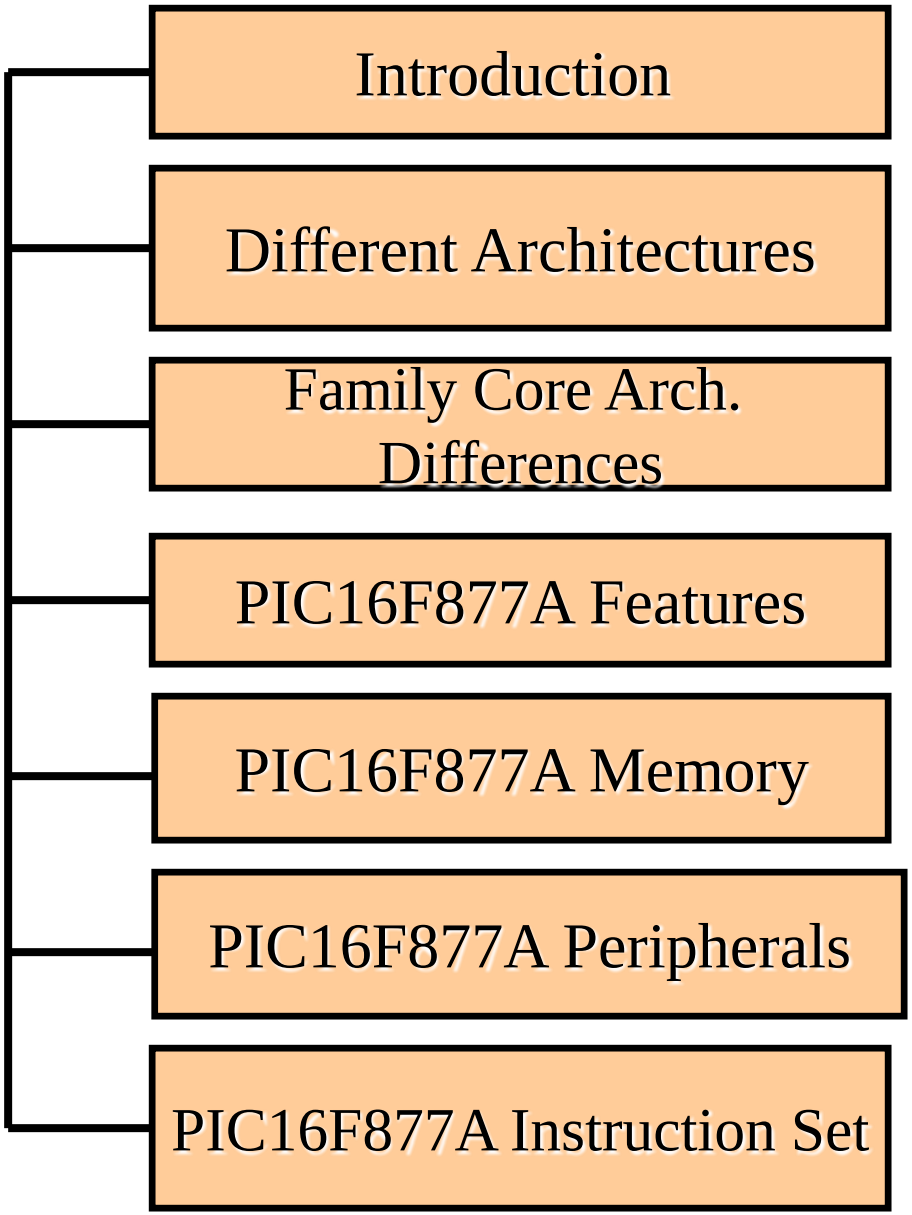
PIC Microcontrollers

Overview


Introduction
Different Architectures
Family Core Arch. Differences
PIC16F877A Features
PIC16F877A Memory
PIC16F877A Peripherals
PIC16F877A Instruction Set

Introduction

- What is PIC?

  - A family of Harvard architecture microcontrollers made by Microchip Technology

  - Derived from the PIC1650 originally developed by General Instrument Microelectronics Division.

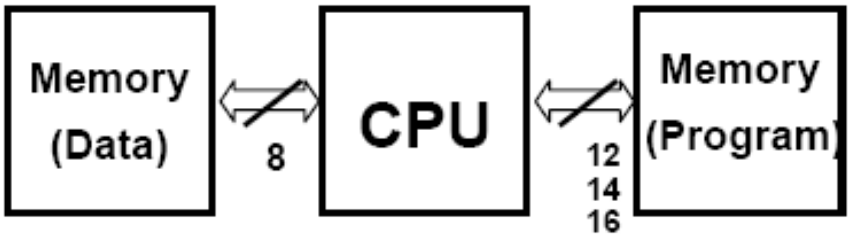
  - The name PIC was originally an acronym for "**Programmable Intelligent Computer**".

Introduction

- Why PIC is popular?
  - low cost ,wide availability with high clock speed
  - availability of low cost or free development tools
  - Only 37 instructions to remember
  - serial programming and re-programming with flash memory capability
  - Its code is extremely efficient, allowing the PIC to run with typically less program memory than its larger competitors
  - PIC is very small and easy to implement for non-complex problems and usually accompanies to the microprocessors as an interface

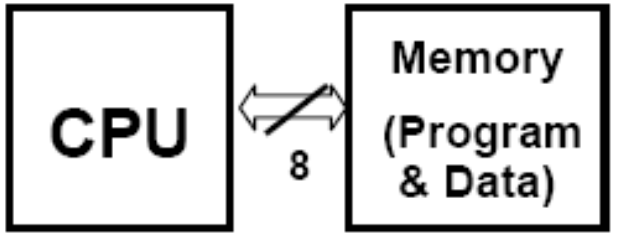
Two Different Architectures

- Harvard Architectures (newer arch.)



Harvard Architecture

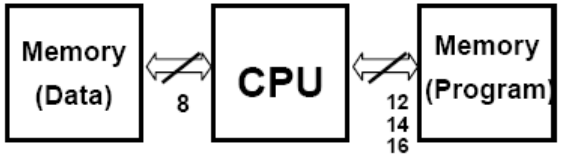
- Von-Neumann Architecture



Von–Neumann Architecture

Two Different Architectures

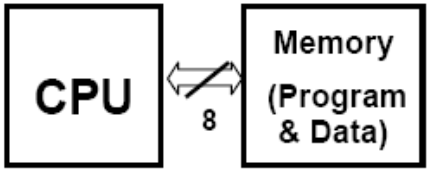
- Harvard Architectures



Harvard Architecture

- Used mostly in RISC CPUs
- Separate program bus and data bus: can be of different widths
- For example, PICs use:
  - Data memory (RAM): a small number of 8bit registers
  - Program memory (ROM): 12bit, 14bit or 16bit wide (in EPROM, FLASH, or ROM)

- Von-Neumann Architecture



Von-Neumann Architecture

- Used in: 80X86 (CISC PCs)
- Only one bus between CPU and memory
- RAM and program memory share the same bus and the same memory, and so must have the same bit width
- **Bottleneck**: Getting instructions interferes with accessing RAM

RISC vs. CISC

- Reduced Instruction Set Computer (RISC)
  - Used in: SPARC, ALPHA, Atmel AVR, etc.
  - Few instructions (usually < 50)
  - Only a few addressing modes
  - Executes 1 instruction in 1 internal clock cycle (Tcyc)
- Complex Instruction Set Computer (CISC)
  - Used in: 80X86, 8051, 68HC11, etc.
  - Many instructions (usually > 100)
  - Several addressing modes
  - Usually takes more than 1 internal clock cycle (Tcyc) to execute

Family Core Architecture Differences

- The PIC Family: Cores
  - 12bit cores with 33 instructions: 12C50x, 16C5x
  - 14bit cores with 35 instructions: 12C67x,16Cxxx
  - 16bit cores with 58 instructions: 17C4x,17C7xx
  - ‘Enhanced’ 16bit cores with 77 instructions: 18Cxxx

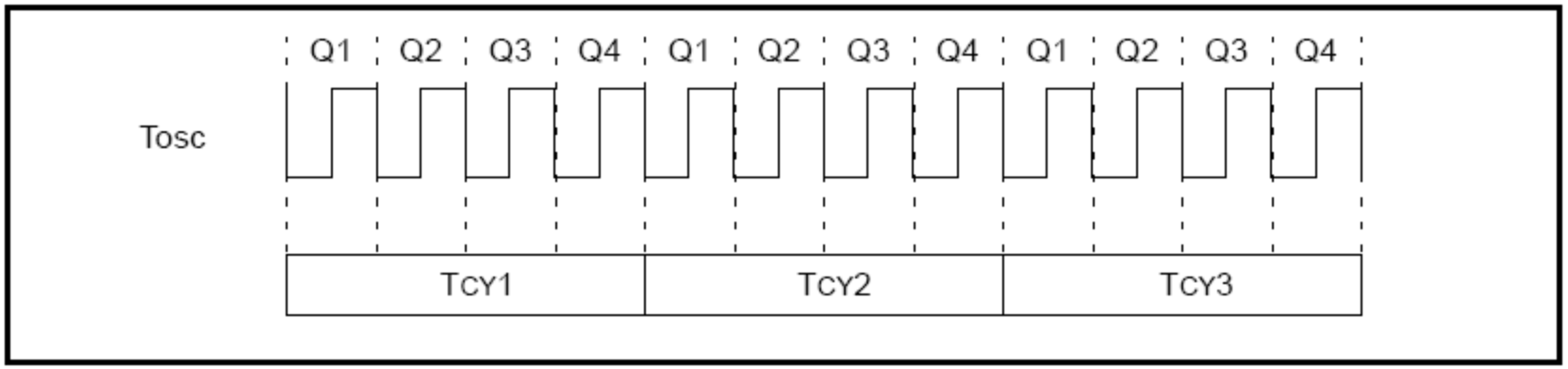
The PIC Family: Speed

- Can use crystals, clock oscillators, or even an RC circuit.
- Some PICs have a built in 4MHz RC clock, Not very accurate, but requires no external components!
- Instruction speed = 1/4 clock speed (Tcyc = 4 * Tclk)
- All PICs can be run from DC to their maximum specified speed:

12C50x	4MHz
12C67x	10MHz
16Cxxx	20MHz
17C4x / 17C7xxx	33MHz
18Cxxx	40MHz

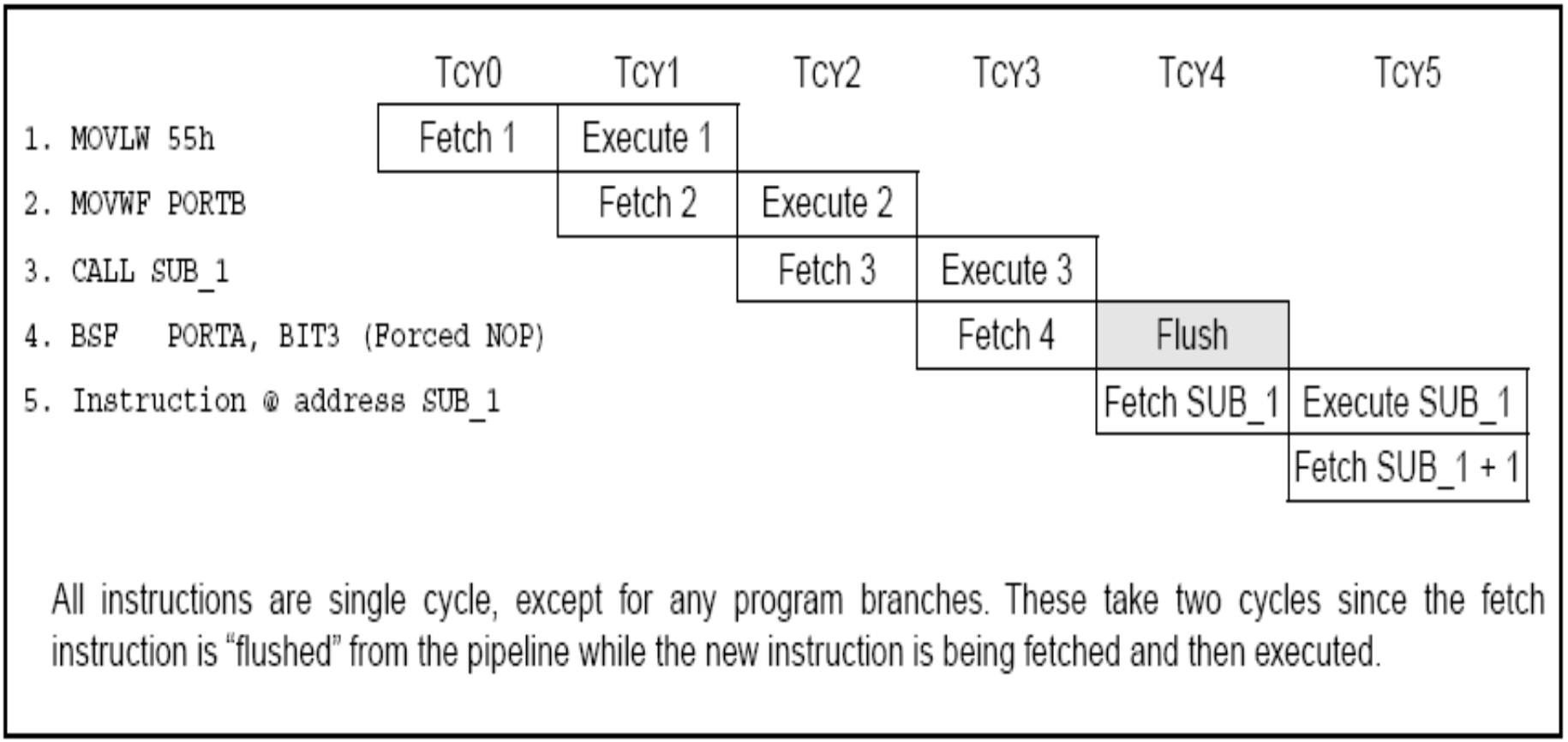
Clock and Instruction Cycles

- Instruction Clock
  - Clock from the oscillator enters a microcontroller via OSC1 pin where internal circuit of a microcontroller divides the clock into four even clocks Q1, Q2, Q3, and Q4 which do not overlap.
  - These four clocks make up one **instruction cycle** (also called machine cycle) during which one instruction is executed.
  - Execution of instruction starts by calling an instruction that is next in string.
  - Instruction is called from program memory on every Q1 and is written in instruction register on Q4.
  - Decoding and execution of instruction are done between the next Q1 and Q4 cycles. On the following diagram we can see the relationship between instruction cycle and clock of the oscillator (OSC1) as well as that of internal clocks Q1-Q4.
  - Program counter (PC) holds information about the address of the next instruction.



Pipelining in PIC

- Instruction Pipeline Flow



All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

The PIC Family: Program Memory

- Technology: EPROM, FLASH, or ROM
- It varies in size from one chip to another.

  - examples:

12C508	**512**	**12bit instructions**
16C711	**1024 (1k)**	**14bit instructions**
16F877	**8192 (8k)**	**14bit instructions**
17C766	**16384 (16k)**	**16bit instructions**

The PIC Family: Data Memory

- PICs use general purpose “File registers” for RAM (each register is 8bits for all PICs)
  - examples:

12C508	25B RAM
16C71C	36B RAM
16F877	368B RAM + 256B of nonvolatile EEPROM
17C766	902B RAM

PIC Programming Procedure

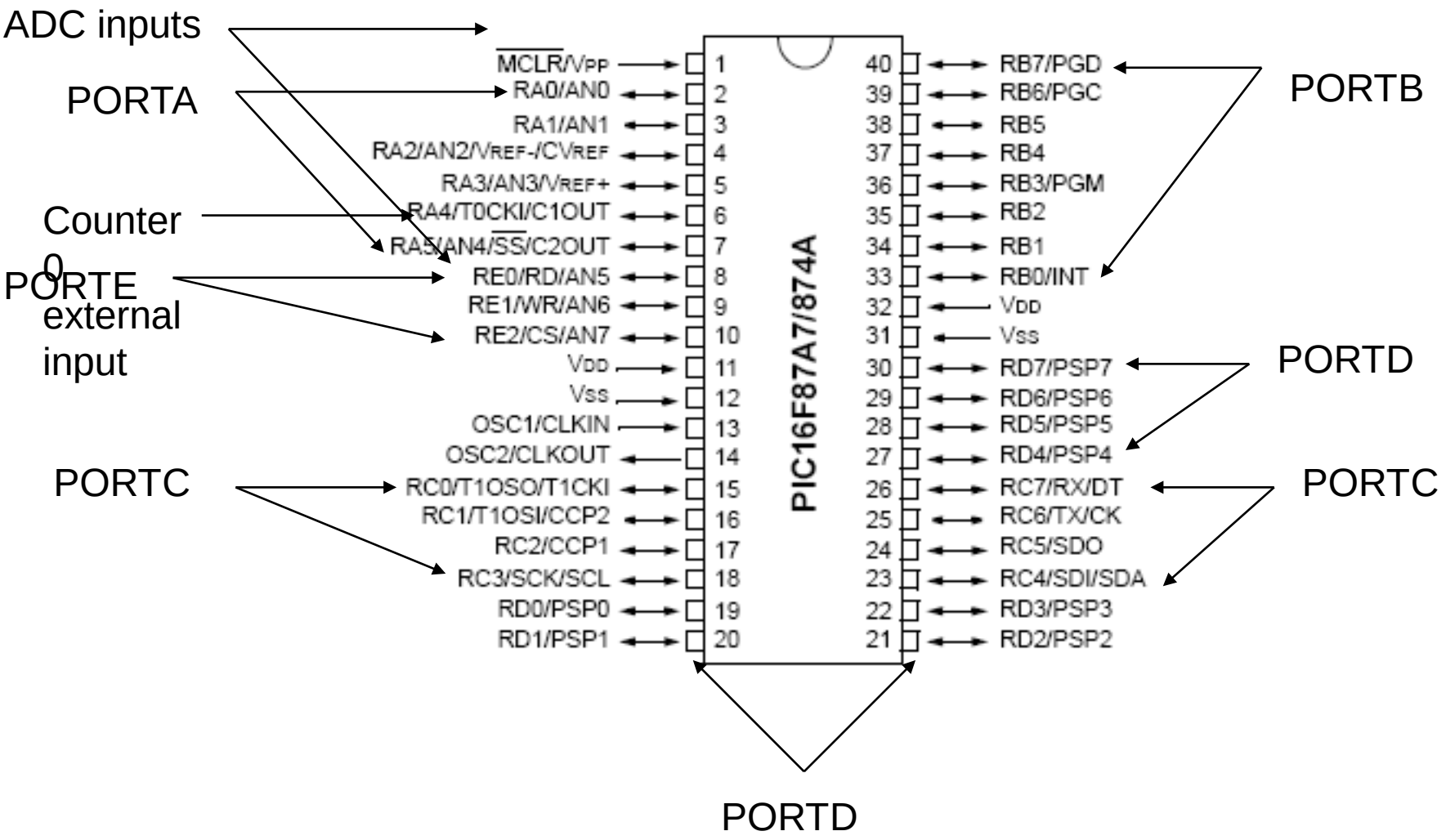
- For example: in programming an embedded PIC featuring electronically erasable programmable read-only memory (EEPROM). The essential steps are:
  - Step 1: On a PC, type the program, successfully compile it and then generate the HEX file.
  - Step 2: Using a PIC device programmer, upload the HEX file into the PIC. This step is often called **"burning"**.
  - Step 3: Insert your PIC into your circuit, power up and verify the program works as expected. This step is often called **"dropping"** the chip. If it isn't, you must go to Step 1 and **debug** your program and repeat burning and dropping.

PIC16F877A Features

High Performance RISC CPU:

- Only 35 single word instructions to learn
- All single cycle instructions except for program branches, which are two-cycle
- Operating speed: DC - 20 MHz clock input DC - 200 ns instruction cycle

PIC16F877A Pin Layout

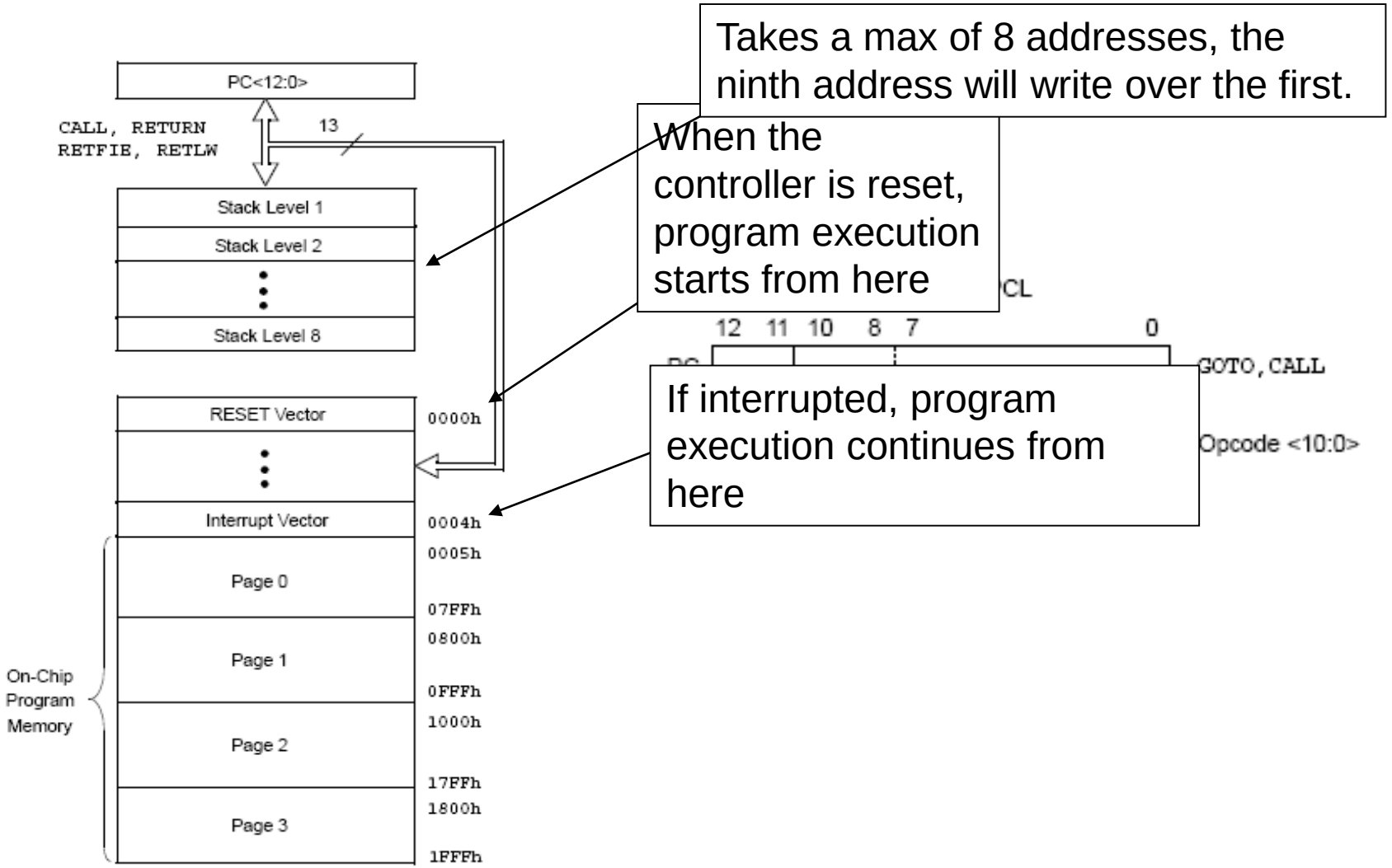


PIC Memory

- The PIC16F877A has an 8192 (8k) 14bit instruction program memory
- 368 Bytes Registers as Data Memory :
  - Special Function Registers: used to control peripherals and PIC behaviors
  - General Purpose Registers: used to a normal temporary storage space (RAM)
- 256 Bytes of nonvolatile EEPROM

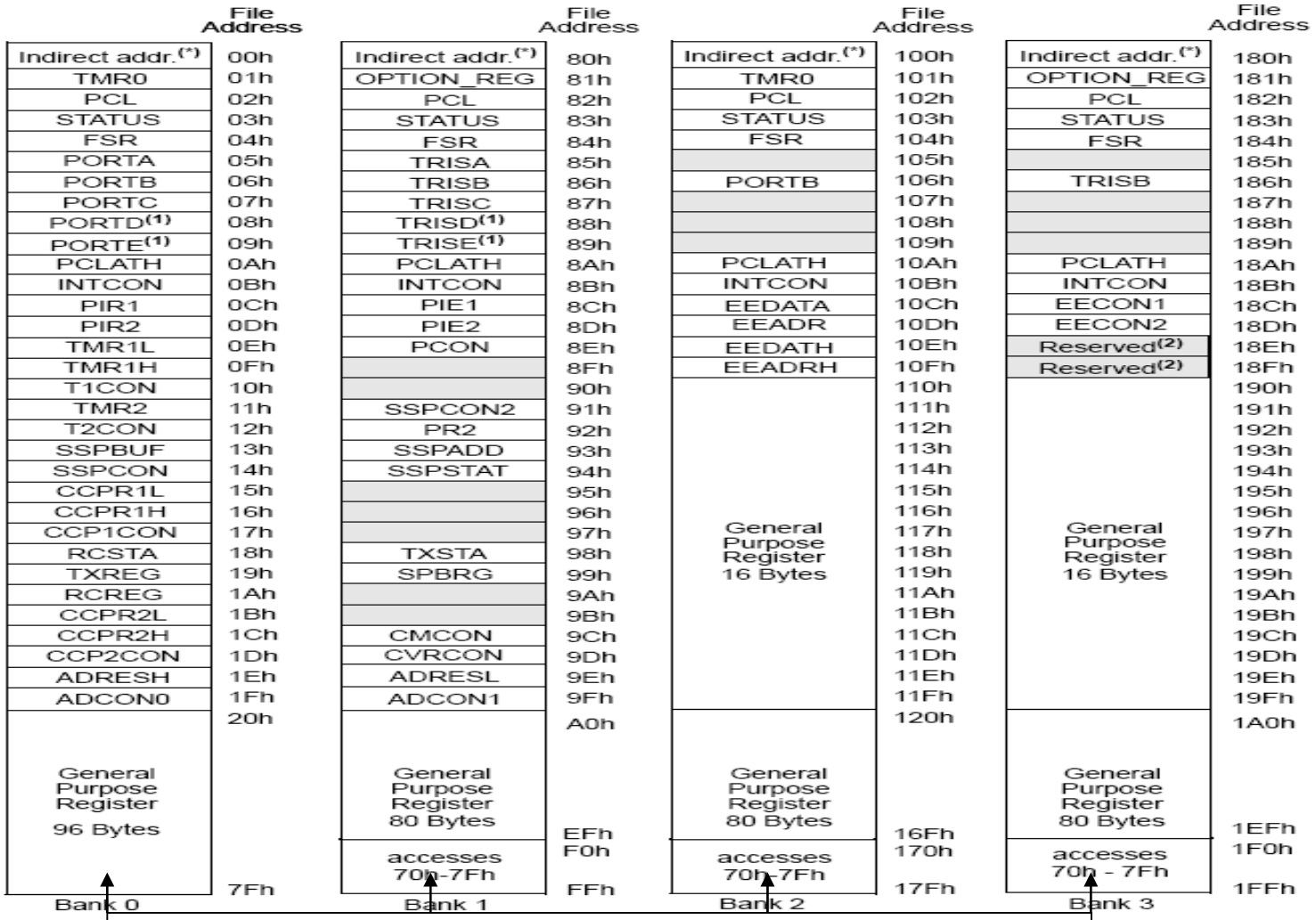
PIC Program Memory

- The PIC16F877 8192 (8k) 14bit instructions



PIC Data Memory

The most important registers have addresses in all the four banks



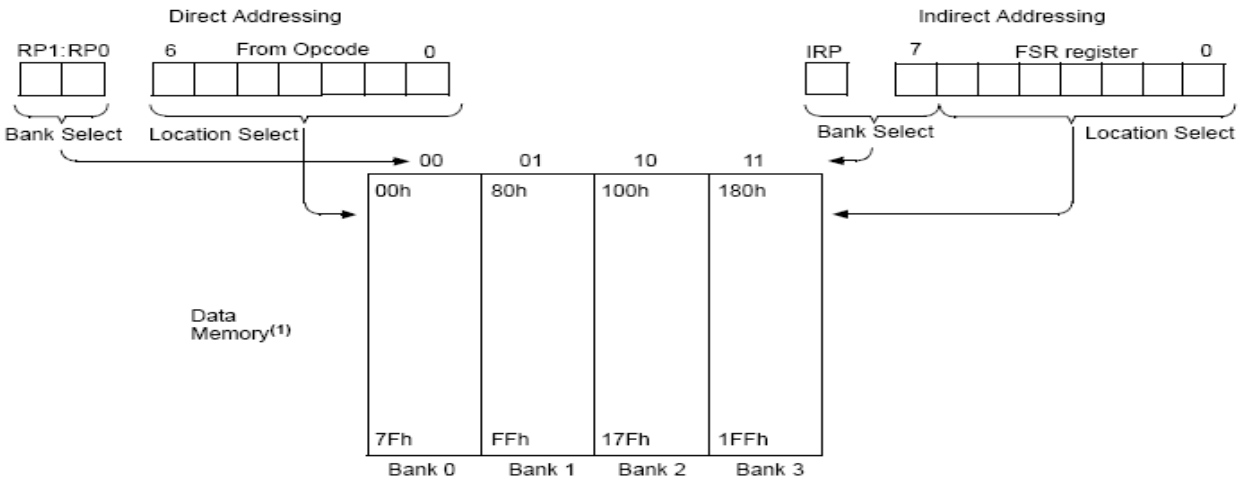
The data memory is devided into 4 memory banks

Note 1: These registers are not implemented on the PIC16F876A.
2: These registers are reserved, maintain these registers clear.

Register Addressing Modes

Immediate Addressing:

Movlw H'0F'



Indirect Addressing:

- Full 8 bit register address is written the special function register FSR
- INDF is used to get the content of the address pointed by FSR
- Exp : A sample program to clear RAM locations H'20' – H'2F:

```
        MOVLW 0x20 ;initialize pointer
        MOVWF FSR ;to RAM
NEXT    CLRF INDF ;clear INDF register
        INCF FSR,F ;inc pointer
        BTFSS FSR,4 ;all done?
        GOTO NEXT ;no clear next
CONTINUE
; ;yes continue
```

Direct Addressing:

PIC Family Control Registers

- Uses a series of “Special Function Registers” for controlling peripherals and PIC behaviors.
  - **STATUS** ➔ Bank select bits, ALU bits (zero, borrow, carry)
  - **INTCON** ➔ Interrupt control: interrupt enables, flags, etc.
  - **OPTION_REG** ➔ contains various control bits to configure the TMR0 prescaler/WDT postscaler ,the External INT Interrupt, TMR0 and the weak pull-ups on PORTB

Special Function Register ”STATUS Register“

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **$\overline{TO}$**: Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 **$\overline{PD}$**: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
(for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{borrow}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Special Function Register "INTCON Register"

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **TMR0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC Peripherals

- Each peripheral has a set of SFRs to control its operation.
- Different PICs have different on-board peripherals

Peripheral Features

- 5 Digital I/O Ports
- Three timer/counter modules
  - Timer0: 8-bit timer/counter with 8-bit pre-scaler
  - Timer1: 16-bit timer/counter with pre-scaler, can be incremented during SLEEP via external crystal/clock
  - Timer2: 8-bit timer/counter with 8-bit period register, pre-scaler and post-scaler
- A 10-bit ADC with 8 inputs
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls

PIC Peripherals: Ports (Digital I/O)

- Ports are basically digital I/O pins which exist in all PICs

- The PIC16F877A have the following ports:
  - PORT A has 6 bit wide, Bidirectional
  - PORT B,C,D have 8 bit wide, Bidirectional
  - PORT E has 3 bit wide, Bidirectional

- Ports have 2 control registers
  - TRISx sets whether each pin is an input (1) or output (0)
  - PORTx sets their output bit levels or contain their input bit levels

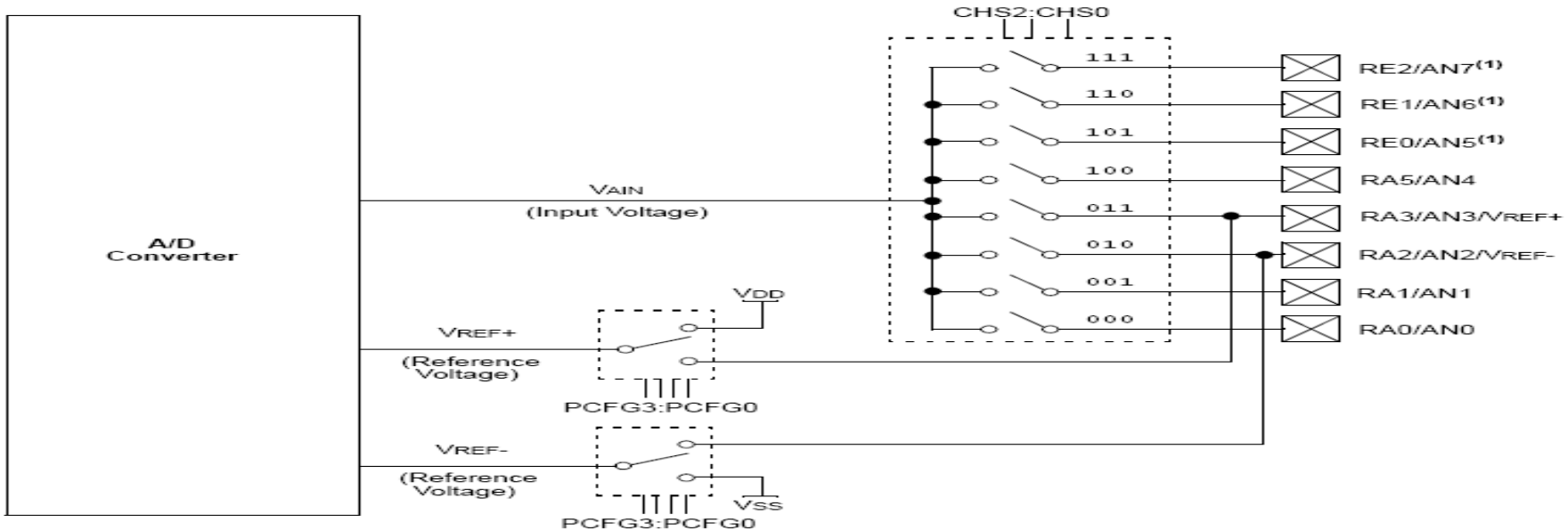
- Pin functionality “overloaded” with other features
- Most pins have 25mA source/sink thus it can drive LEDs directly

PIC Peripherals: Analogue to Digital Converter

- Only available in 14bit and 16bit cores
- Fs (sample rate) < 54KHz
- the result is a 10 bit digital number
- Can generate an interrupt when ADC conversion is done

PIC Peripherals: Analogue to Digital Converter

- The A/D module has four registers. These registers are:
  - A/D Result High Register (ADRESH)
  - A/D Result Low Register (ADRESL)
  - A/D Control Register0 (ADCON0)
  - A/D Control Register1 (ADCON1)
- Multiplexed 8 channel inputs
  - Must wait T_{acq} to charge up sampling capacitor
- Can take a reference voltage different from that of the controller



PIC Peripherals: USART: UART

- Serial Communications Peripheral:

Universal Synch./Asynch. Receiver/Transmitter

- Interrupt on TX buffer empty and RX buffer full

- Asynchronous communication: UART (RS-232C serial)
  - Can do 300bps - 115kbps
  - 8 or 9 bits, parity, start and stop bits, etc.
  - Outputs 5V so you need a RS232 level converter (e.g., MAX232)

PIC Peripherals: USART: UART

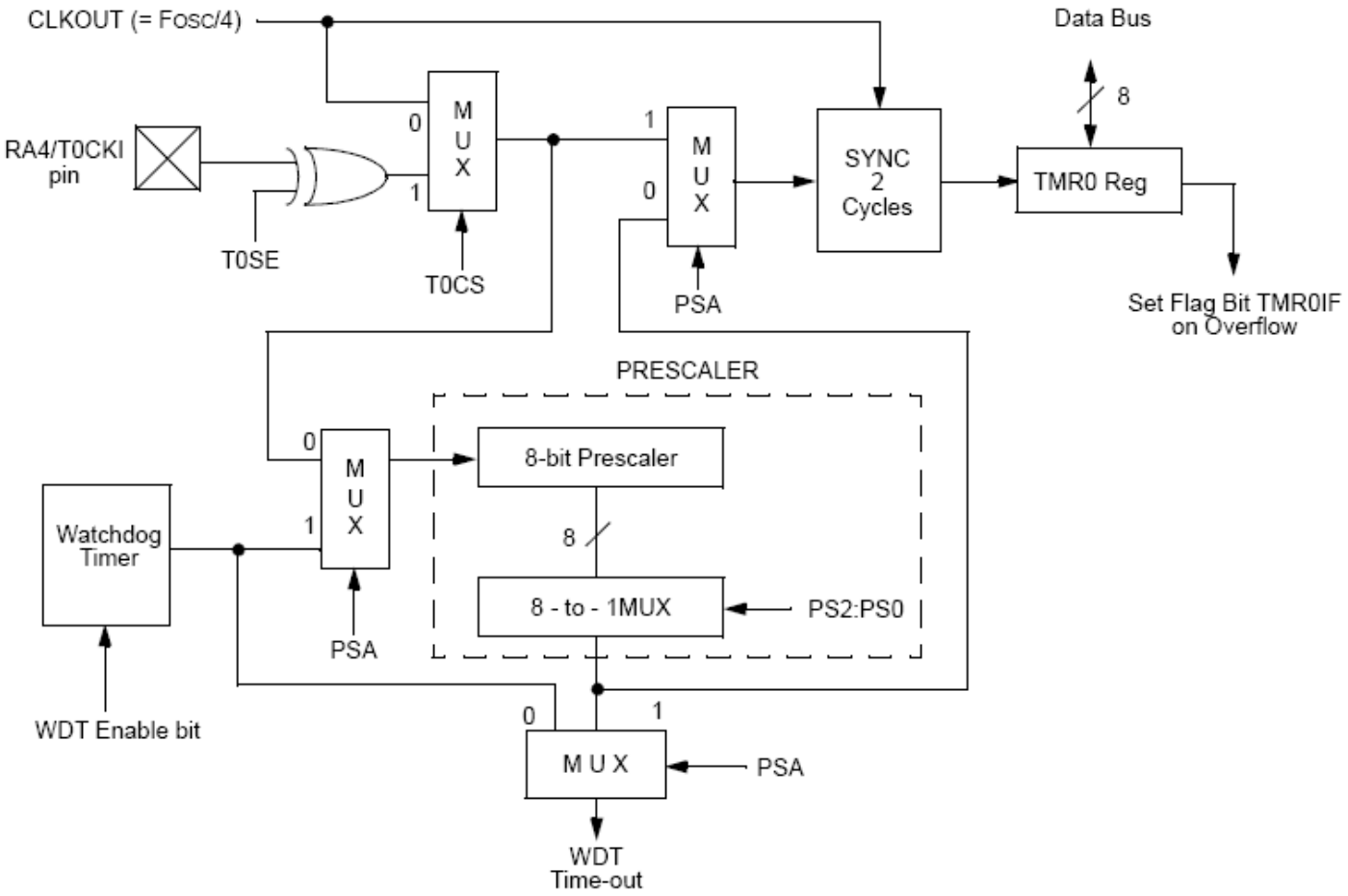
- Synchronous communication: i.e., with clock signal
- SPI = Serial Peripheral Interface
  - 3 wire: Data in, Data out, Clock
  - Master/Slave (can have multiple masters)
  - Very high speed (1.6Mbps)
  - Full speed simultaneous send and receive (Full duplex)
- I2C = Inter IC
  - 2 wire: Data and Clock
  - Master/Slave (Single master only; multiple masters clumsy)
  - Lots of cheap I2C chips available; typically < 100kbps

PIC Peripherals: Timers

- Available in all PICs.
- generate interrupts on timer overflow.
- Some 8bits, some 16bits, some have prescalers and/or postscalers
- Can use external pins as clock in/clock out

(ie, for counting events or using a different Fosc)

Timer 0 Block Diagram



Special Function Register
OPTION_REG Register

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

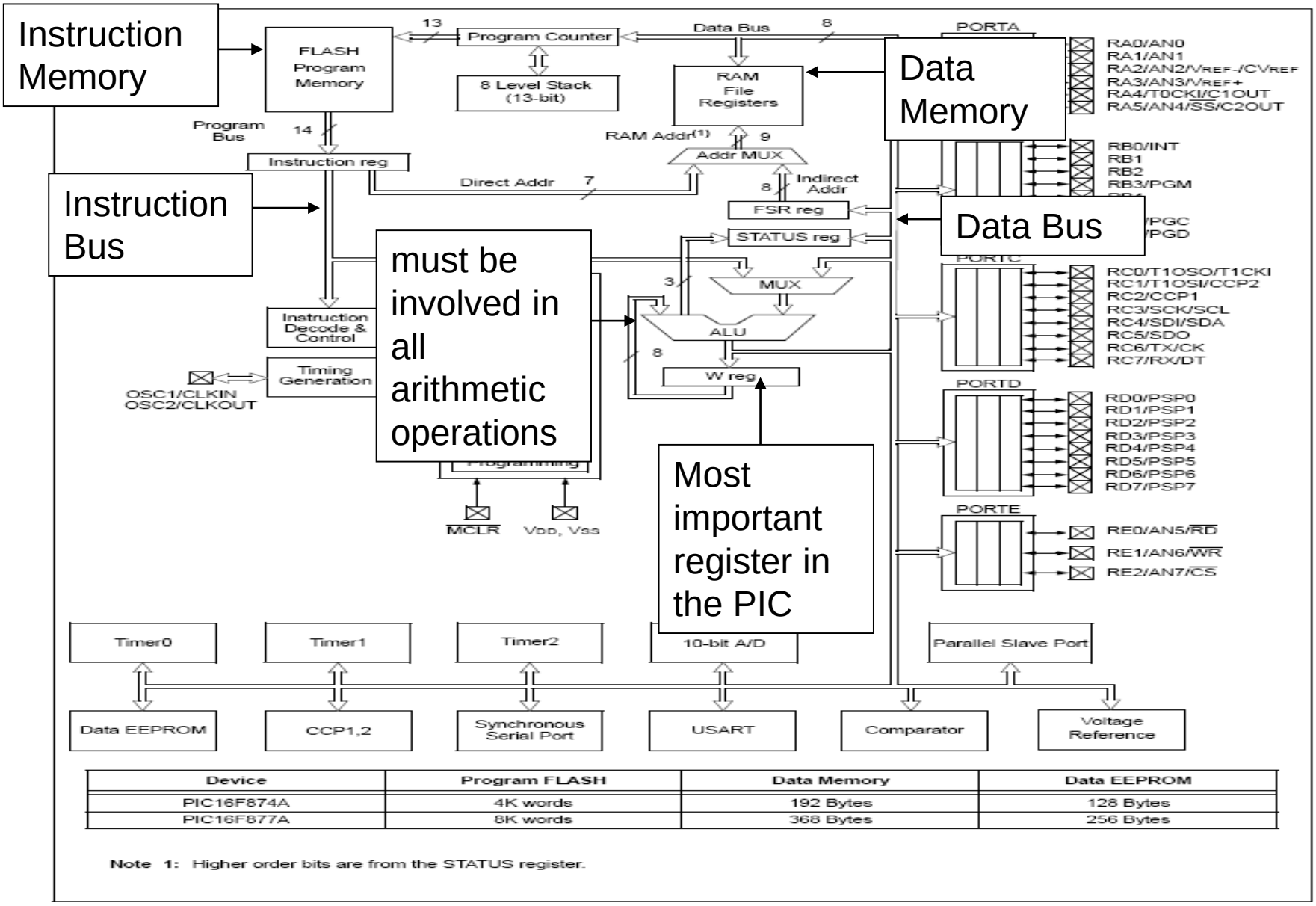
bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

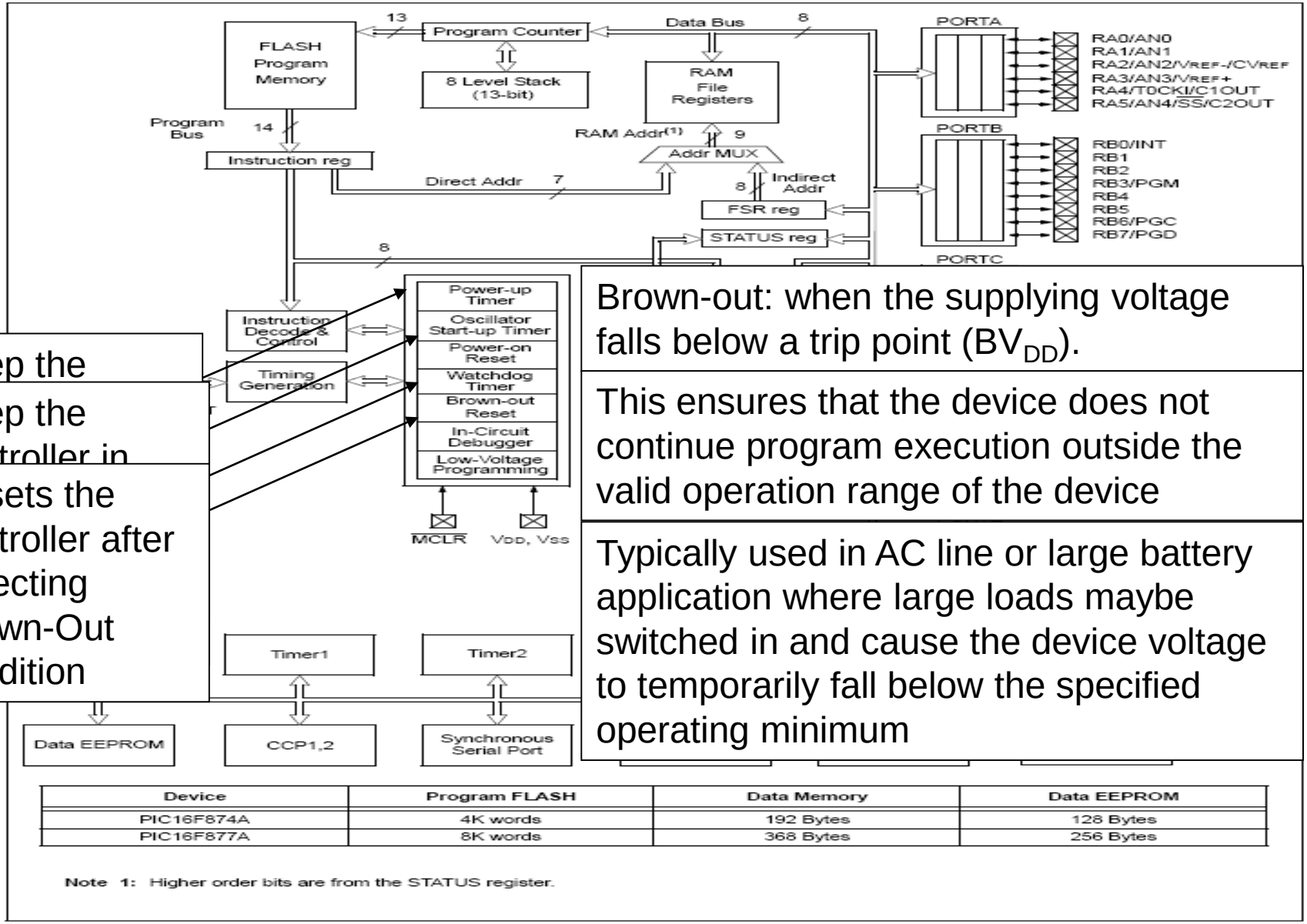
PIC16F877A Block Diagram



Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the STATUS register.

PIC16F877A Block Diagram



Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the STATUS register.

Keep the

Keep the controller in

Resets the controller after detecting Brown-Out condition

Brown-out: when the supplying voltage falls below a trip point (BV_{DD}).

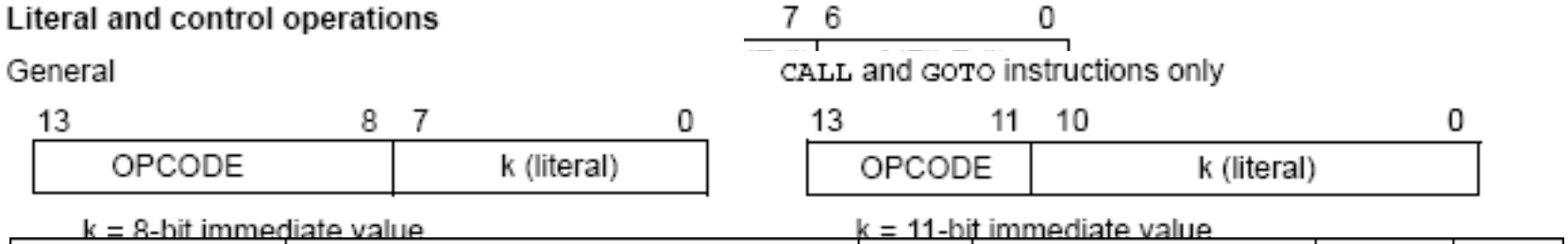
This ensures that the device does not continue program execution outside the valid operation range of the device

Typically used in AC line or large battery application where large loads maybe switched in and cause the device voltage to temporarily fall below the specified operating minimum

PIC16F877A Instruction Set

Bit-oriented file register operations

Literal and control operations



ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2

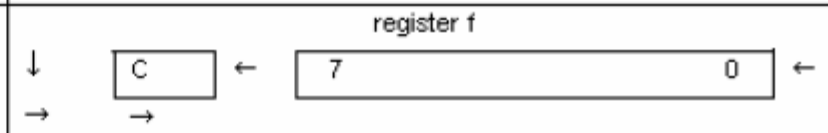
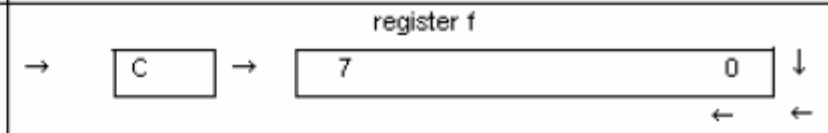
Literal and Control Instructions

Mnemonic		Description	Function
addlw	k	Add literal to W	k + W → W
andlw	k	AND literal and W	k .AND. W → W
call	k	Call subroutine	PC + 1 → TOS, k → PC
clrwdt		Clear watchdog timer	0 → WDT (and prescaler if assigned)
goto	k	Goto address (k is nine bits)	k → PC (9 bits)
iorlw	k	Incl. OR literal and W	k .OR. W → W
movlw	k	Move Literal to W	k → W
option		Load OPTION register	W → OPTION Register
retfie		Return from Interrupt	TOS → PC, 1 → GIE
retlw	k	Return with literal in W	k → W, TOS → PC
return		Return from subroutine	TOS → PC
sleep		Go into Standby Mode	0 → WDT, stop oscillator
sublw	k	Subtract W from literal	K - W → W
tris	f	Configure port f (downward compat. instr.)	W → I/0 control reg f
xorlw	k	Exclusive OR literal and W	k .XOR. W → W

Key:

Field	Description		
b	Bit address within an 8-bit file register		
d	Destination select;	d = 0	Store result in W
		d = 1	Store result in file register f.
			Default is d = 1.
f	Register file address (0x00 to 0xFF)		
k	Literal field, constant data or label		
W	Working register (accumulator)		

Byte-Oriented Instructions

Mnemonic		Description	Function
addwf	f,d	Add W and f	W + f → d
andwf	f,d	AND W and f	W .AND. f → d
clrf	f	Clear f	0 → f
clrw		Clear W	0 → W
comf	f,d	Complement f	.NOT. f → d
decf	f,d	Decrement f	f - → d
decfsz	f,d	Decrement f, skip if zero	f - 1 → d, skip if 0
incf	f,d	Increment f	f + 1 → d
incfsz	f,d	Increment f, skip if zero	f + 1 → d, skip if 0
iorwf	f,d	Inclusive OR W and f	W .OR. f → d
movf	f,d	Move f	f → d
movwf	f	Move W to f	W → f
nop		No operation	
rlf	f,d	Rotate left f	
rrf	f,d	Rotate right f	
subwf	f,d	Subtract W from f	f - W → d
swapf	f,d	Swap halves f	f(0:3) ↔ f(4:7) → d
xorwf	f,d	Exclusive OR W and f	W .XOR. f → d

Bit-Oriented Instructions

Mnemonic		Description	Function
bcf	f,b	Bit clear f	0 → f(b)
bsf	f,b	Bit set f	1 → f(b)
btfsc	f,b	Bit test, skip next instruction if clear	skip if f(b) = 0
btfss	f,b	Bit test, skip next instruction if set	skip if f(b) = 1

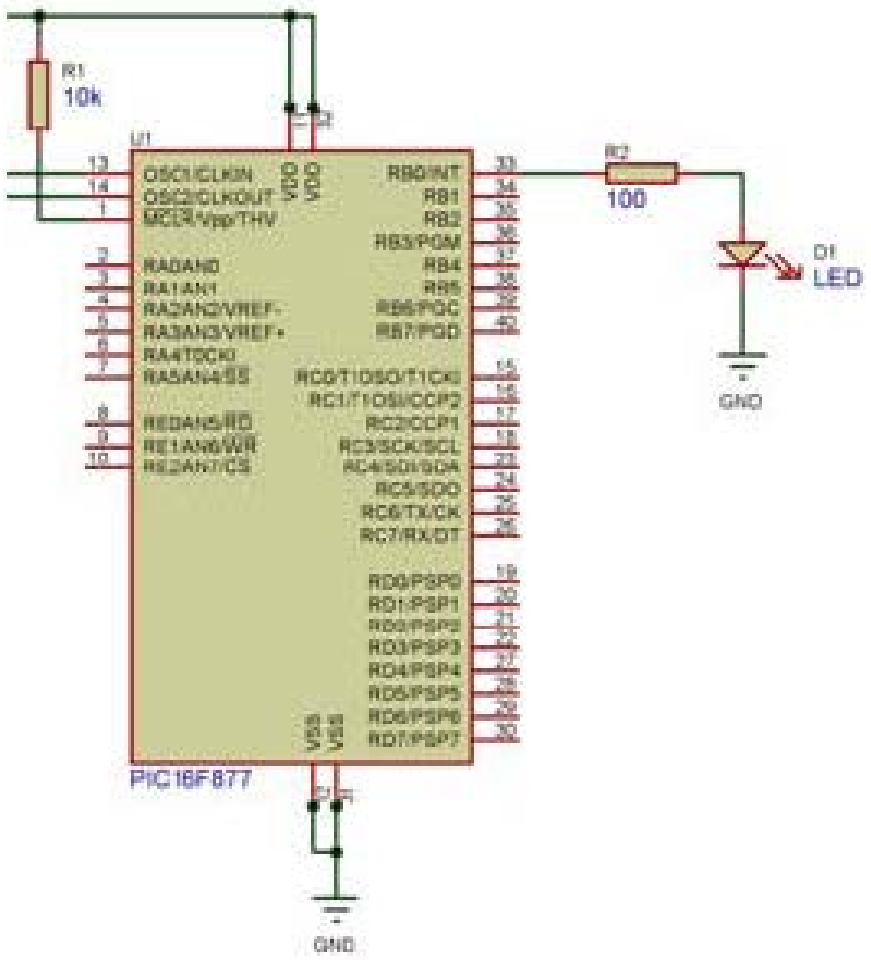
Key:

Field	Description		
b	Bit address within an 8-bit file register		
d	Destination select;	d = 0	Store result in W
		d = 1	Store result in file register f.
			Default is d = 1.
f	Register file address (0x00 to 0xFF)		
k	Literal field, constant data or label		
W	Working register (accumulator)		

PIC Applications

- LED Flasher

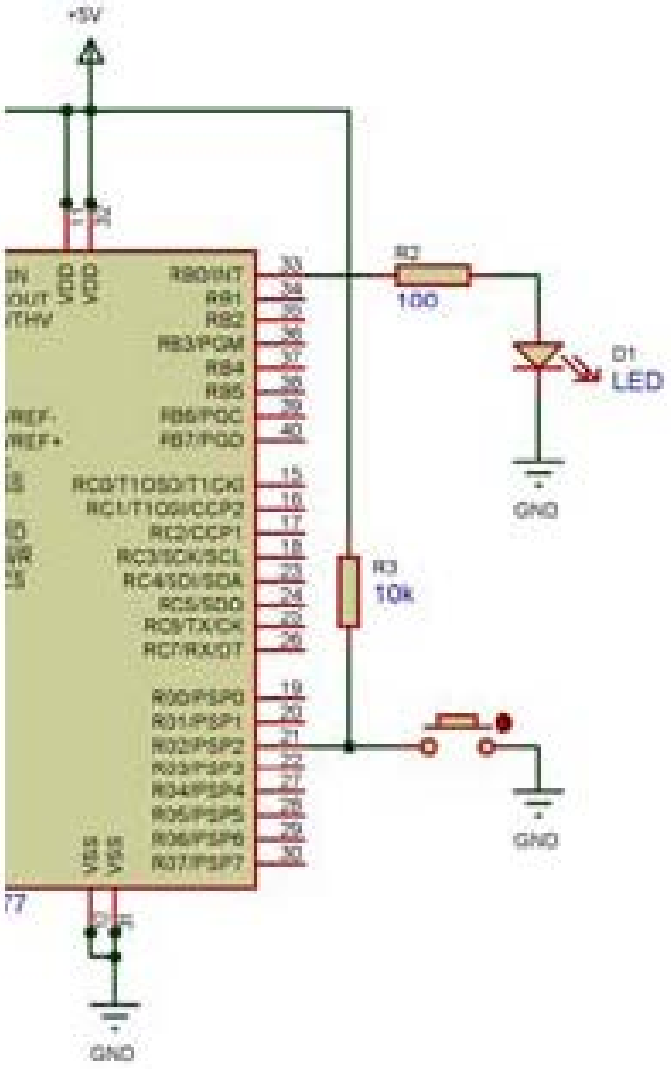
```
Loop:
        bsf     PORTB, 0
        call    Delay_500ms
        bcf     PORTB, 0
        call    Delay_500ms
        goto    Loop
```



PIC Applications

- Button Read

```
            Movlw   0
            movwf   TRISD, f
            bsf     TRISD, 2
Loop:
            btfsc   PORTD, 2
            goto    light
            goto    No_light
Light:
            bsf     PORTB,0
            goto    Loop
No_light:
            bcf     PORTB,0
            goto    Loop
```



References and Further Readings

- http://www.microchip.com
- http://en.wikipedia.org/wiki/PIC_microcontroller
- 16F87x Data Sheet
- Mid Range Manual

Thank You For Your Attendance.